Digital representations

Contribution to Educational Activities of IAPR Technical Committee on Discrete Geometry (TC18)

Isabelle Bloch
Ecole Nationale Supérieure des Télécommunications - CNRS UMR 5141 LTCI
Paris - France - Isabelle.Bloch@enst.fr

January 5, 2005

Contents

1 Introduction

Computerized image processing requires digital representations of the observed scenes. We have to deal with a sampling of $\mathbb{R}^2$ or $\mathbb{R}^3$ in $\mathbb{Z}^2$ or $\mathbb{Z}^3$ (depending on whether the images are two-dimensional or three-dimensional), which is moreover finite. For this reason, images are often represented as matrices. In order to deal with such sets of points, two solutions can be envisaged:

- either the digital points of $\mathbb{Z}^n$ are embedded in $\mathbb{R}^n$, and they can be treated as points of a continuous space (methods from algorithmical geometry are then often involved for their processing);
- or objects and operations are directly defined in the digital space, by trying to keep as much as possible their effects and properties.

The choice of one of these solutions depends a lot on the type of processing to be performed, on the type of representation on which this processing relies, on the required properties, on the computational complexity, etc. For instance, basic morphological operators are easier to perform in digital spaces. In cases the two approaches are possible, they are not always equivalent, and it is often difficult to guarantee that the transformation applied to a digital version of an object will provide the same result as the digitization of the result of the continuous transformation.

In this presentation[1], we are mainly interested in digital geometrical representations, which will serve as a support to operations and processing directly performed in $\mathbb{Z}^n$. We briefly address the issue of tessellations and grids in Section 2, on which digital images are defined. Topology on such representations is the focus of Section 3. Issues related to the representation of a few simple geometrical structures are addressed in Section 4. An introduction to digital distances is provided in Section 5. More about these last topics can be found in the relevant sections of TC18[2]. A detailed presentation of all aspects related to digital geometry can be found in [5] (in French), or in [15] for instance.

The main principles introduced in this presentation can be applied to image analysis and to image synthesis as well. Digital geometry still constitutes an important research area in both domains.

[1]This text is for the main part a translation of a lecture I give at ENST in Paris, as an introduction to digital representations in a course dedicated to image processing.

[2]`http://www.cb.uu.se/~tc18/educational.html`

2 Tessellations and grids

2.1 Definitions and constraints

A tessellation is a partition of the continuous space (typically $\mathbb{R}^n$) composed of elementary cells (the whole space is covered by cells, two cells are not intersecting).

This most general definition can give rise to an infinity of solutions if no additional constraints are imposed on the shape and spatial arrangement of the cells. From a practical point of view, constraints can be imposed:

- on the one hand by the sensors, which have a sensing surface which has generally a regular structure;
- and on the other hand by the subsequent use of the representation, which may impose some regularity and simplicity, as well as some properties on the resulting tessellation, that will be detailed below.

Two dual methods can be explored to build such a tessellation:

1. The first method consists in defining a distribution of points in space. To each point P a cell V_P is associated, in such a way that it does not intersect any other cell and does not leave any empty space. The easiest solution consists in defining V_P as the set of points which are closer to P than to any other point of the distribution. In the case of a regular distribution, a classical tessellation is obtained, as the ones described in the next section. In the case of an irregular distribution, a Voronoï tessellation is obtained, that will be shortly described in Section 4.5. The points of the distribution can be interpreted as seeds from which cells grow at constant speed until the space is completely filled (see Figure 1).

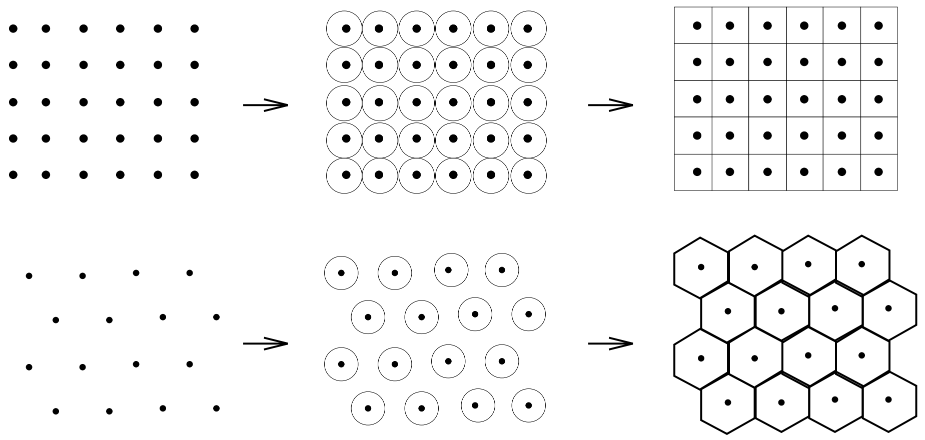


Figure 1: Two examples of tessellation construction through propagation from a regular point distribution.

2. The second solution consists in choosing an a priori model for an elementary cell and reproducing this model by arranging cells so as to build a partition of space [11]. This is obviously not possible with any cell model. Usually it is imposed that the model is a convex regular polygon or polyhedron and that the vertices are in contact with other vertices only. An example of configuration where these requirements are not matched is illustrated in Figure 2.

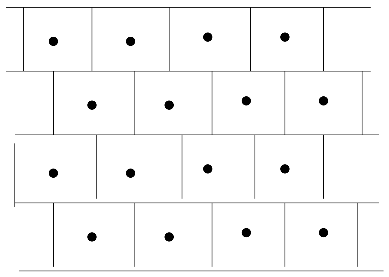


Figure 2: Example of a configuration where vertices of the tessellation do not coincide with other vertices. Note that this point distribution can lead to an appropriate tessellation by assigning an hexagon to each point instead of a rectangle.

2.2 Regular planar tessellations

The most strict constraint that can be imposed to elementary cells is that they are all identical and regular, in the sense that all edges of the polygons have the same length, all angles are equal, and every vertex is connected to a constant number of vertices of other cells. Such elementary cells provide so called *regular tessellations*.

It can be shown that, in the Euclidean plane, there exist only three types of regular tessellations, corresponding respectively to triangular, square and hexagonal cells. They are illustrated in Figure 3.

Triangular tessellations are regular in the sense of the above definition, but they are not uniform, since two different orientations of the triangles occur in the tessellation, as can be observed in Figure 3.

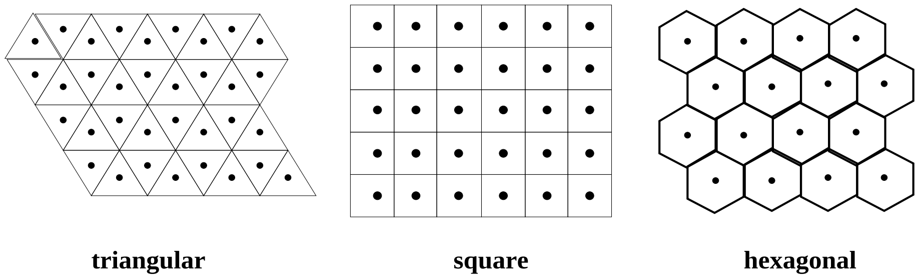


Figure 3: The three types of regular tessellations in the Euclidean plane.

2.3 Semi-regular planar tessellations

Now if the previous constraints are somewhat relaxed, we may allow cells to be of different types, with different numbers of edges. When keeping a constant number of adjacent cells at each vertex, we obtain *semi-regular tessellations* [11].

There exist 21 solutions satisfying these constraints. However, only 11 of them lead actually to a partition of space (among them are the three regular ones). Two of these solutions are illustrated in Figure 4. The complete set of solutions can be found in [5].

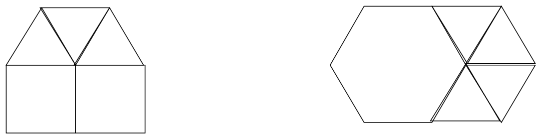
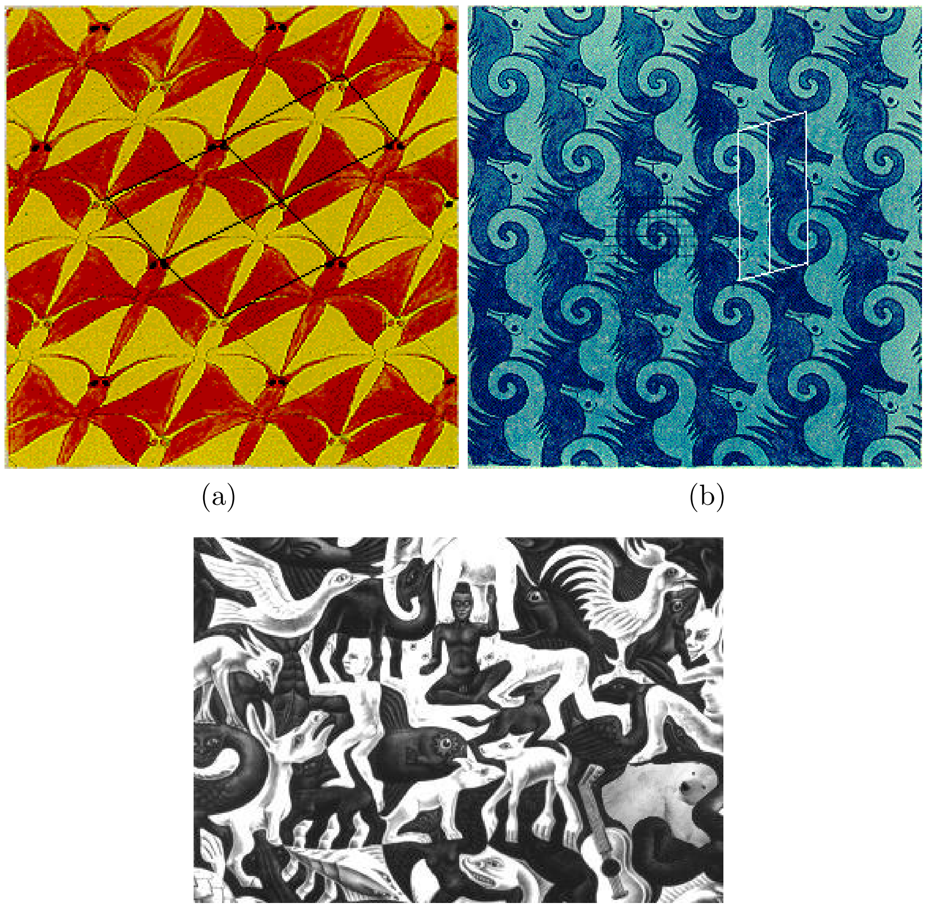


Figure 4: Two examples of semi-regular configurations leading to planar tessellations.

2.4 Non-regular tessellations

If even more constraints are relaxed, non uniform of even completely irregular tessellations can be obtained (see for instance the well known drawings by Escher [18], such as the examples in Figure 5).

We are not going into further details here on the associated theory.



(a) (b) (c)

Figure 5: Examples of Escher's tessellations, based on an underlying structure and symmetry properties (a and b), or with cells that are all different (c).

2.5 Duality between tessellation and mesh

We may assign to each cell V_P of a tessellation a point P inside this cell. For instance, P can be chosen as the center of V_P (which belongs to V_P for convex cells). Then the points P and cells

V_P correspond to the construction of tessellations from point distributions. These points P are the pixels in images.

A mesh is then defined as follows: for each point P, we determine the points Q such that V_P and V_Q are adjacent along an edge (in plane). The mesh is constituted by all segments $[P, Q]$ obtained this way.

This mesh defines a new tessellation in space, which cells centered on the vertices of the initial tessellation. This exhibits a duality between tessellation and mesh. The term "grid" is also often used.

For regular tessellations, which are by far the most popular ones, this duality takes the following form (see Figure 6):

- the mesh associated to the square tessellation is square;
- the mesh associated to the hexagonal tessellation is triangular;
- the mesh associated to the triangular tessellation is hexagonal.

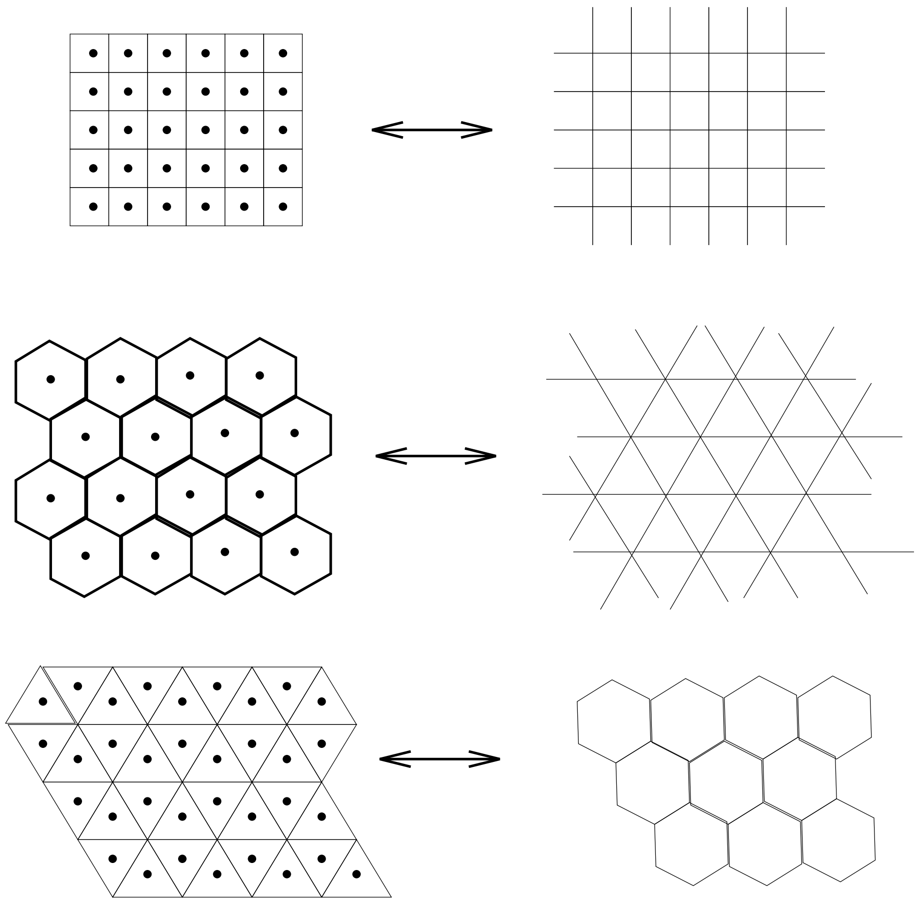


Figure 6: Duality between meshes and regular tessellations of the Euclidean plane.

The square grid is undoubtedly the most used one. The hexagonal grid in also very interesting since it has better isotropy and topology properties than the square grid. Moreover, it corresponds to more efficiency in the sampling process leading to digital images. The triangular mesh is almost never used.

In the digital 3 dimensional space, the cubic grid is the most used one. It is also the only one which is regular. A few applications use cubic grids with additional vertices on the center of the faces, or rhombododecahedric grids.

3 Digital topology

In this section, we address a few topological issues on digital meshes. In particular, it will be shown that classical digital topology based on points is not appropriate and an approach based on definition of neighborhood better suits the needs of digital image processing.

3.1 A few approaches

Let us first consider the classical digital topology, where each point defines an open set. Since a connected set is a set that cannot be decomposed into a union of two non-empty and non-intersecting open sets, the only connected sets that can be built with this topology are reduced to singletons.

This is clearly not very satisfactory, since for image processing and pattern recognition purposes, objects have to be defined as spatial entities containing usually several points and for which connectedness can be an important property. The classical digital topology does not lead to an appropriate representation of such objects and does not match the image processing requirements.

Several approaches have been explored in order to define topologies which are mode appropriate. One of them is now briefly described [4, 13]. It consists in defining a topological basis of $\mathbb{Z}^2$. A topology is then a class $\mathcal{T}$ of subsets of $\mathbb{Z}^2$ such that each element of $\mathcal{T}$ can be expressed as a union of elements of the topological basis, and such that:

1. $\mathbb{Z}^2 \in \mathcal{T}$, $\emptyset \in \mathcal{T}$;
2. $\forall (A, B) \in \mathcal{T}^2,\ A \cap B \in \mathcal{T}$.

On a square grid, a topological basis can be for instance constructed as follows: to each point P of $\mathbb{Z}^2$, with spatial coordinates (i, j), a set $\mathcal{U}(P)$ is associated, defined by (see Figure 7):

$$\mathcal{U}(P) = \begin{cases} \{P, (i-1, j), (i+1, j), (i, j-1), (i, j+1)\} & \text{if } (i+j) \text{ is even,} \\ \{P\} & \text{otherwise.} \end{cases}$$

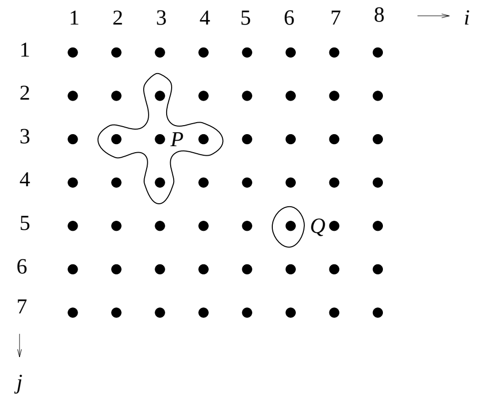


Figure 7: Definition of a topological basis on the digital plane on a square grid.

The sets $\mathcal{U}(P)$ satisfy the following conditions:

1. $\mathbb{Z}^2 = \cup_{P \in \mathbb{Z}^2} \mathcal{U}(P)$;

2. if $\mathcal{U}(P)$ and $\mathcal{U}(Q)$ are two elements of the topological basis, then $\mathcal{U}(P) \cap \mathcal{U}(Q)$ is a union of elements of this basis.

This type of definition has a major drawback: the shape of the elementary neighborhood induced by $\mathcal{U}(P)$ around each point depends on the position of the point in space. In particular, the topology is not invariant under translation. Moreover, this definition does not share all required properties, in particular the Jordan theorem does not hold. Finally, if this construction is still possible on a triangular grid (see Figure 8), it is not possible on an hexagonal grid.

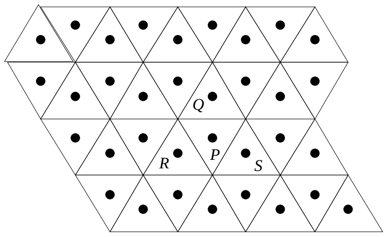


Figure 8: Definition of a topological basis in the digital plane on a triangular grid. In this Figure, $\mathcal{U}(P) = \{P, Q, R, S\}$, $\mathcal{U}(Q) = \{Q\}$, $\mathcal{U}(R) = \{R\}$, $\mathcal{U}(S) = \{S\}$.

3.2 Topology from elementary neighborhood

As shown above, it is difficult to define topological notions which are convenient and operational for image processing using a classical construction based on open sets. Since one of the most important notions for image interpretation and pattern or object recognition is connectivity [24, 26], it is better to rely directly on this type of concept and define directly elementary neighborhood and connectivity in the digital case.

An interesting and powerful approach towards this aim is to consider a digital image as a graph. The vertices of the graph are the digital points (pixels in the image) and the edges are defined through neighborhood relations between points. For instance, on a square grid, we can consider that two points are neighbors if exactly one of their coordinates differs by one unit. Each point (circled in Figure 9 a) has therefore fours neighbors. This defines the *4-connectivity*. The corresponding graph is illustrated in Figure 10.

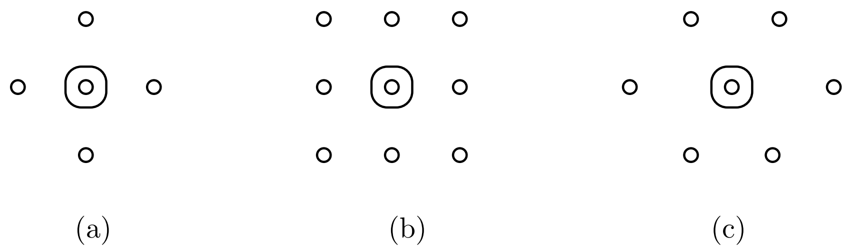


Figure 9: Connectivity on a square digital grid (a – 4-connectivity, b – 8-connectivity), and on an hexagonal grid (c – 6-connectivity).

If edges linking points in diagonal are added to the graph (i.e. points that have both coordinates differing by one unit), then each point has eight neighbors, this defining the *8-connectivity* (see Figure 9 b). Using the 4-connectivity, very few directions of space are represented, while with the 8-connectivity there are more neighbors (and directions) but which are not at the same distance of the central point.

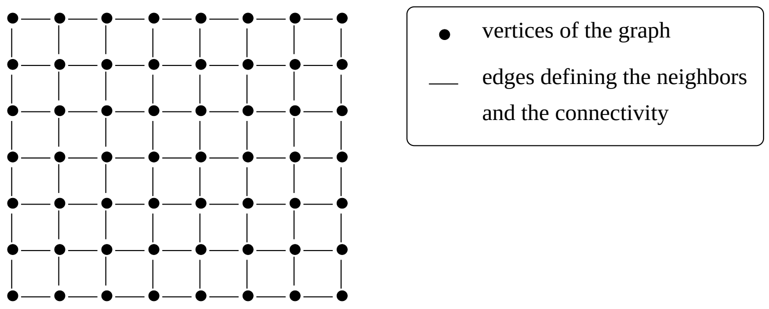


Figure 10: Graph associated to the 4-connectivity on a square grid.

On an hexagonal grid, the most natural connectivity is the *6-connectivity* (see Figure 9 c). Each point has six neighbors which are all at the same distance of the central point. The six represented directions are equally distributed. These features are additional advantages of the hexagonal grid.

On a triangular grid, three types of elementary neighborhood are defined, corresponding to the 3-, 9- and 12-connectivity. These definitions are illustrated in Figure 11.

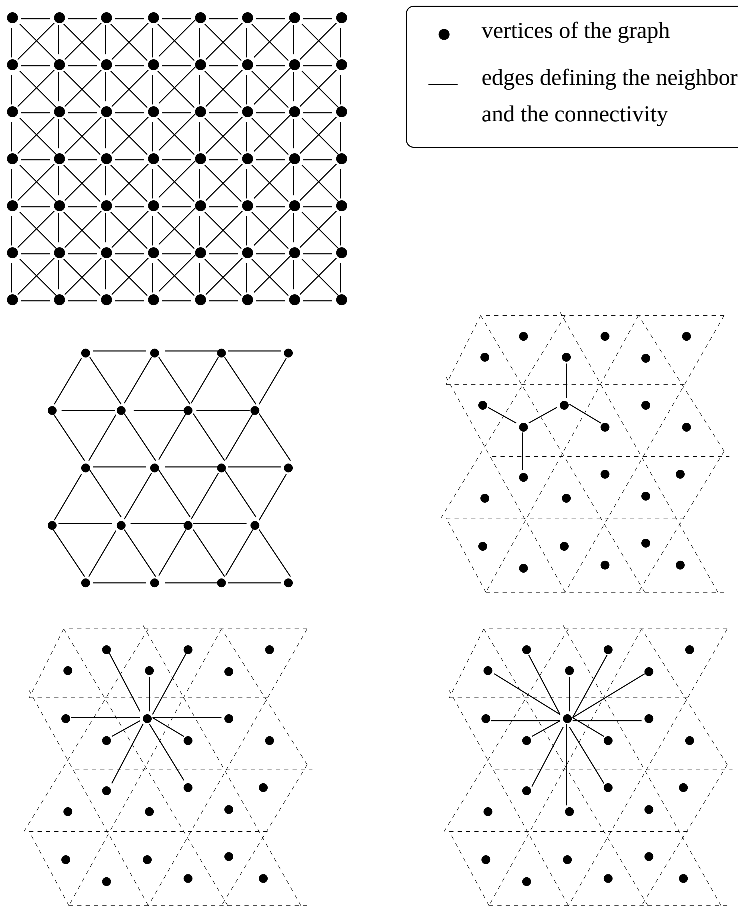


Figure 11: Graph defining the 8-connectivity on a square grid, the 6-connectivity on an hexagonal grid, and 3-, 9- and 12-connectivity on a triangular grid (only a few edges are represented in the last cases).

In a matrix representation of the images, which is the most usual one, accessing the neighbors of a point with coordinates (i, j) is very easy on a square grid. These are:

- in 4-connectivity: points with coordinates $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$;
- in 8-connectivity: points with coordinates $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$, $(i-1, j-1)$, $(i-1, j+1)$, $(i+1, j-1)$, $(i+1, j+1)$.

On an hexagonal grid, accessing the neighbors is somewhat less straightforward. It is still possible to encode the image as a matrix, but geometrically the lines should be considered as shifted with respect to each other by half the distance between two pixels. The indices of the neighbors of a point therefore depend on whether a line number is odd or even. If lines are numbered as indicated in Figure 12, the neighbors of a point with coordinates (i, j) are:

- if j is even: $(i-1, j-1)$, $(i, j-1)$, $(i-1, j)$, $(i+1, j)$, $(i-1, j+1)$, $(i, j+1)$;
- if j is odd: $(i, j-1)$, $(i+1, j-1)$, $(i-1, j)$, $(i+1, j)$, $(i, j+1)$, $(i+1, j+1)$.

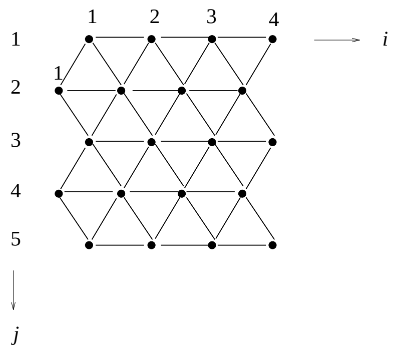


Figure 12: The indices of points on an hexagonal grid depend on whether the number of the line on which they lie is odd or even.

In a similar way, on a triangular grid, the indices of the neighbors depends on the number of the line on which the considered point is lying (and on the considered connectivity).

Let us move now to the 3D case, on a cubic grid. Three types of elementary neighborhood are classically defined, corresponding respectively to the 6-, 18- and 26-connectivity. Let us represent a point P of the 3D image as an elementary cube. The 6-neighbors are the cubes which are adjacent to P by face, the 18-neighbors additionally include those adjacent by edge, and the 26-neighbors those adjacent by vertex (see Figure 13).

Once these elementary neighborhoods are defined, the notions of connectivity can be derived in a very simple way thanks to the analogy with graph theory [2].

A *path* is defined as a sequence of vertices of the graph such that any two consecutive points in the sequence are linked by an edge of the graph, the edges being defined according to the chosen digital connectivity. For instance, on a square grid, we define two types of paths:

- a 4-path (or 4-connected path) is a sequence of points $(i_k, j_k)_{1 \leq k \leq n}$ such that:

$$\forall k, 1 \leq k < n, \ |i_k - i_{k+1}| + |j_k - j_{k+1}| \leq 1;$$

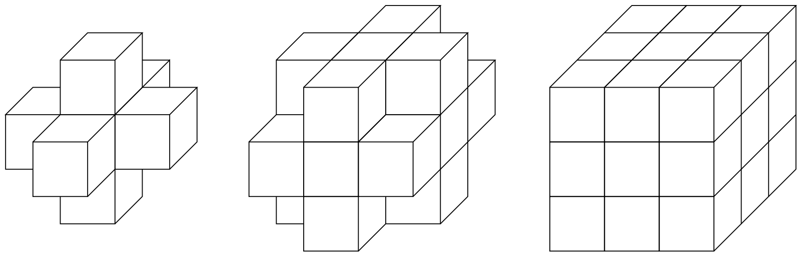


Figure 13: The three types of elementary neighborhood in a 3D image defined on a cubic grid.

- a 8-path (or 8-connected path) is a sequence of points $(i_k, j_k)_{1 \leq k \leq n}$ such that:

$$\forall k, 1 \leq k < n,\ \max(|i_k - i_{k+1}|, |j_k - j_{k+1}|) \leq 1.$$

Connected components are defined from paths as in classical graph theory. On a square grid, definitions are as follows:

- a 4-connected component is a set of points $\mathcal{S}$ such that for every pair of points (P, Q) in $\mathcal{S}$, there exists a 4-connected path with end points P and Q and included in $\mathcal{S}$;
- a 8-connected component is a set of points $\mathcal{S}$ such that for every pair of points (P, Q) in $\mathcal{S}$, there exists a 8-connected path with end points P and Q and included in $\mathcal{S}$.

These definitions are illustrated in Figure 14.

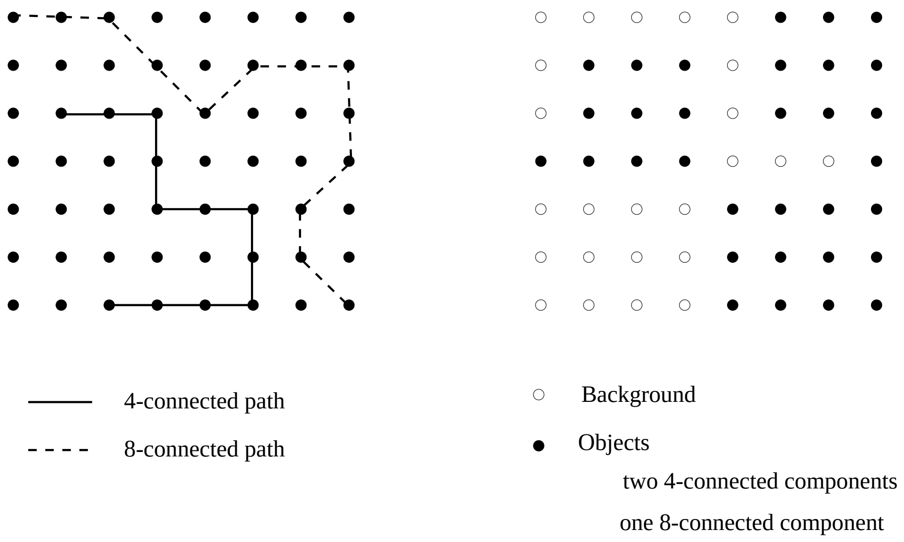


Figure 14: Illustration of path and connected component definitions on a square grid, in 4- and 8-connectivity.

On an hexagonal grid, we define analogously 6-connected paths and 6-connected components. On a triangular grid, paths and connected components can be 3-, 9- or 12-connected.

On a square grid, paradoxes can occur for some local configurations of points of an object and its complement. In Figure 15, if 8-connectivity is considered, points a and d of the object are neighbors, and so are points b and c of the complement of the object (the object points are represented by crosses while the background points are represented by circles). This means that connected components of the object and of its complement are somehow “crossing” each other, which contradicts the intuition. On the contrary, if 4-connectivity is chosen, neither a and d nor b and c are neighbors, thus leaving an empty area which is neither in the object nor in its complement, which contradicts again the common sense. This paradox can be avoided if different connectivities are considered for the object and its complement.

On an hexagonal grid, this type of phenomenon does not occur, and the same connectivity (6-connectivity) can be used both for objects and background.

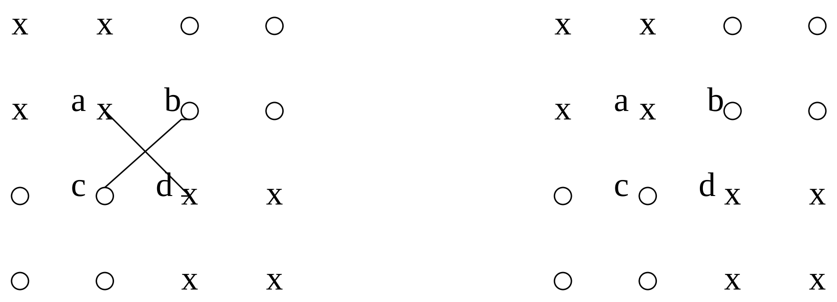


Figure 15: Connectivity paradox on a square grid with 4- and 8-connectivity.

These paradoxes can be explained, and properly solved, using *Jordan’s theorem*, which is a very important result for guaranteeing good topological properties. In the continuous case, it says: every simple closed curve separates the space into two connected components, the interior and the exterior of the curve (a closed curved is a curve with no extremities, i.e. a loop, and it is called simple if it does not self-intersect). Based on the definitions of digital connectivity given above, the two following results can be proved:

- on a square grid, every 4-connected path (respectively 8-connected path) which is closed and simple[3] separates the digital space into two 8-connected components (respectively 4-connected components), the interior and the exterior of the digital curve (see the illustration in Figure 16);
- on an hexagonal grid, every simple closed 6-connected path separates the digital space into two 6-connected components.

These results are digital equivalents of Jordan’s theorem.

The first result demonstrates the duality between 4-connectivity and 8-connectivity [16]. Indeed, in order to have Jordan’s theorem, the connectivity has to change when moving from the object (the digital curve in the theorem) to its complement. Therefore we will either consider 4-connected objects and 8-connected background, or the opposite. In Figure 16, the points of the interior of the 4-connected path form a 8-connected component, but two 4-connected components, illustrating that the theorem does not hold if the same connectivity is used for the path and its complement.

[3]A 4-connected path $(A_0, ..., A_n)$ is simple and closed if $n \geq 4$, $A_i = A_j$ iff $i = j$, and A_i is a 4-neighbor of A_j iff $i = j \pm 1[n + 1]$.

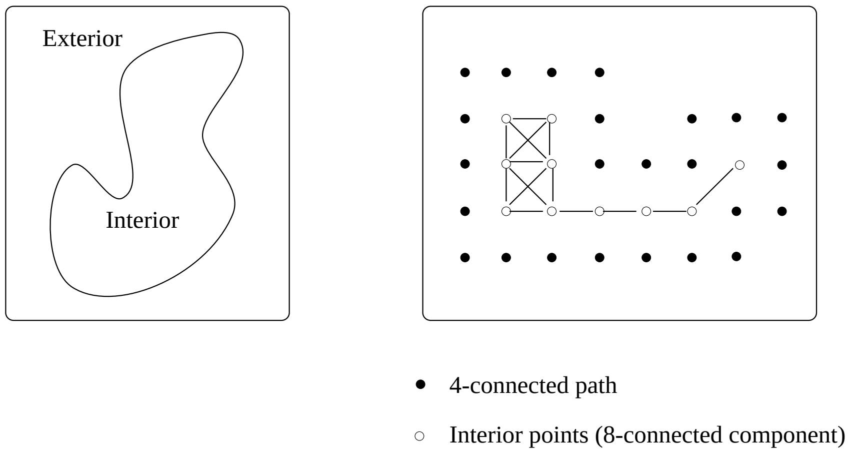


Figure 16: Illustration of Jordan's theorem in the continuous case and in the digital case on a square grid (exterior points are not represented).

The second result shows that this type of problem cannot occur on an hexagonal grid with the 6-connectivity. This grid has thus better topological properties than the square grid, leading also to a simpler processing.

On a triangular grid, analogous results can be proved, exhibiting a duality between 3- and 12-connectivities. However, as already mentioned, this grid is of reduced interest and rarely used.

In 3D on a cubic grid, there is a duality between 6- and 26-connectivities and an extension of Jordan's theorem holds, by considering a closed simple surface.

The difficulty we encountered with 4- and 8-connectivities is due to the lack of clear membership of the boundary of a pixel to the object or its complement. If we consider that a pixel corresponds to a square cell in space, the edges and vertices of this cell are adjacent to other cells, which raises an ambiguity when two adjacent cells belong to two different structures in the image. This ambiguity can be solved by using more precise structures, such as cellular complexes [17]. Each pixel (or voxel in 3D) is decomposed into elements of local dimension varying from 0 (vertices) to 2 (interior of a cell) or 3 (interior of a cube representing a voxel). The membership of each element is explicitly defined. These structures do not raise any topological problem and Jordan's theorem naturally holds, as illustrated in Figure 17.

These structures can be extended to 3D [6].

Despite their good topological properties and the very convenient representation they provide for objects that may have parts of different local dimensions, cellular complexes have two drawbacks: this representation has a higher memory cost, and it lacks uniformity, since elements of different types have to be managed, along with different types of neighborhoods.

3.3 Euler number: an example of topological characteristic of an objet

An object or set of points can be characterized by several topological measures. The simplest ones are the number of connected components of the object and the number of holes it contains. The difference between these two numbers is called *Euler number* and can be used in pattern recognition problems as one of the features characterizing an object.

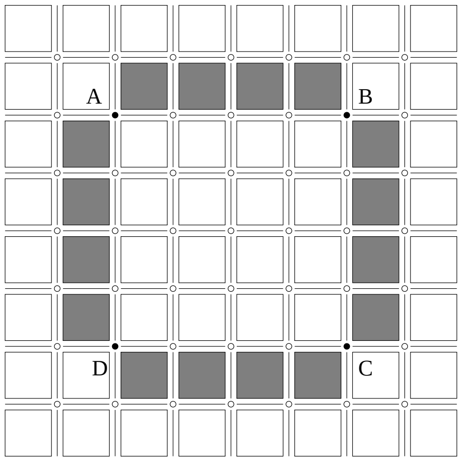


Figure 17: Jordan's theorem is always satisfied in the model of cellular complexes. Each pixel is decomposed into an interior square, edges and vertices. The "color" of points A, B, C and D determines the connnectivity of the black curve and of the background.

Let us denote by N_{cc} the number of connected components of an object and by N_h the number of holes it contains, then the Euler number is defined as $E = N_{cc} - N_h$. The connectivity used for computing the number of holes and the number of connected components has to satisfy the constraints expressed in the digital version of Jordan's theorem. The Euler number depends on the choice of the connectivity. In the example displayed in Figure 18, it is equal to -1 if objects are considered in 8-connectivity and holes (complement of the objects inside the object) in 4-connectivity, and to 0 if the opposite conventions are adopted.

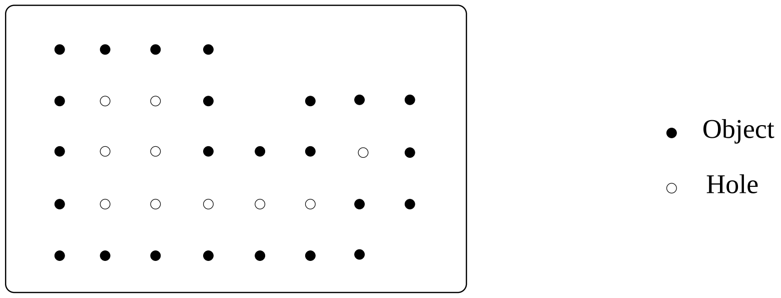


Figure 18: Euler number on a square grid: if objects are considered in 8-connectivity and holes in 4-connectivity, $N_{cc} = 1$ and $N_h = 2$, thus $E = -1$. With the opposite conventions, $N_{cc} = 1$ and $N_h = 1$, thus $E = 0$.

The Euler number can be computed by a simple enumeration of local configurations, described in Figure 19. With the notations of this Figure, it is shown that:

- if objects are considered in 8-connectivity and holes in 4-connectivity, then:

$$E = v - e - d + t - q;$$

- if objects are considered in 4-connectivity and holes in 8-connectivity, then:

$$E = v - e + q.$$

In these equations, v is obtained by counting the configurations of type v, etc. It is clear that a point can be involved in several configurations.

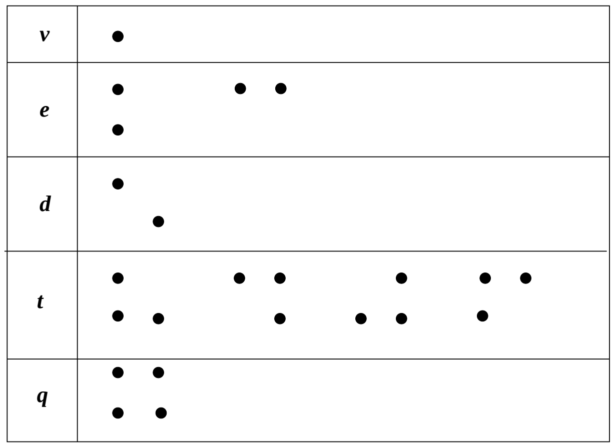


Figure 19: Local configurations involved in the computation of the Euler number.

It is easy to verify that applying these formulas on the example in Figure 18 provides the expected results.

A noticeable feature of these equations and of the configurations is that they only involve object points, while the Euler number involves both the object and its complement. This example is also interesting because it shows that it is possible to compute the Euler number only by counting local configurations, although it is a global feature of the objects. The computation is therefore very easy and the implementation with these local configurations is straightforward.

This aspect is quite common in image processing, and a lot of algorithms try to implement global notions with only local computations. This is an additional advantage of the notion of elementary neighborhoods, since the points they define are the ones involved in local computations.

4 Representations of some geometrical entities

While most usual and simple geometrical entities, such as lines, circles, curves, are easy to handle in the continuous Euclidean space, their digital representations raise two types of problems:

- how could a continuous geometrical entity be represented on a digital grid, and what would be the properties of this representation in comparison to those that hold in the continuous case?
- given a digital geometrical entity, what are the corresponding continuous representations?

Let us consider the example of a line. It is possible to design rules (as will be seen below) which allow us to determine the points of the grid representing a continuous line. Conversely, given a set of points on the grid, it is possible to check whether they correspond or not to the digitization of a continuous line according to these rules. If this is the case, there is in general not a unique line yielding to this set of points, but a set of possible lines.

A study of such problems for general curves can be found e.g. in [5, 15]. Here we focus only on lines and circles. Then we briefly describe Voronoï tessellations and Delaunay triangulations, which lead to structural representations of images, not in a systematic way as with regular grids, but as a function of the image content.

In what follows, we consider the 2D case, and a square grid. The mentioned results can in general be extended to other types of grids and to higher dimension spaces.

4.1 Digitization of a continuous line

A first method for digitizing lines relies on the notion of "semi-open square". A semi-open square associated with a digital point P with coordinates (i, j) is the set of points of $\mathbb{R}^2$ having coordinates (x, y) which satisfy:

$$i - \frac{1}{2} < x \leq i + \frac{1}{2} \text{ and } j - \frac{1}{2} < y \leq j + \frac{1}{2}.$$

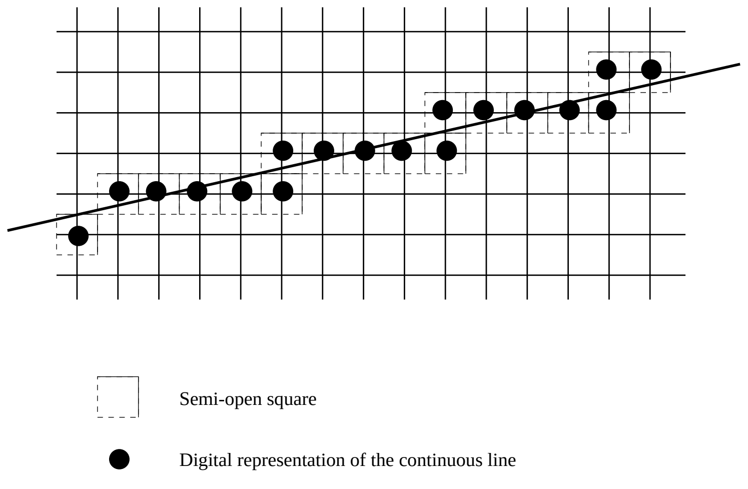


Figure 20: Digitization of a line segment with the semi-open square method.

The digitization process then consists in keeping the digital points P on the grid such that the semi-open square associated to P has a non-empty intersection with the entity to be digitized. Figure 20 illustrates the result in the case of a line segment.

The digital set obtained this way does not necessarily constitute a simple path in the sense of the 4- or 8-connectivity. On the example in Figure 20, the obtained set of points is not 4-connected, and if 8-connectivity is considered, some points have more than 2 neighbors.

However, if the half-plane defined by the line is digitized and then its boundary is taken as the digital line (instead of digitizing directly the line), a 8-connected path can be obtained, by imposing a uni-laterality constraint [8]. The method consists in keeping the digital points situated on the same side of the line and such that the unit vertical segment originating in each point intersects the continuous line. This procedure is illustrated in Figure 21.

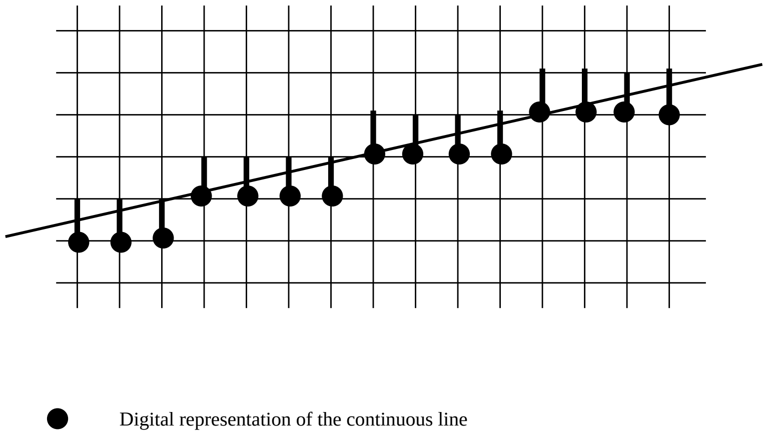


Figure 21: Digitization of a line segment with a uni-laterality constraint.

Let us note that the obtained result differs from the one obtained with the semi-open square method (for the same continuous line).

It is possible to combine both approaches by associating to each grid point a semi-open segment, centered at this point and vertical. A digital point is then kept if the segment originating in it intersects the continuous line. The result of this procedure is illustrated in Figure 22.

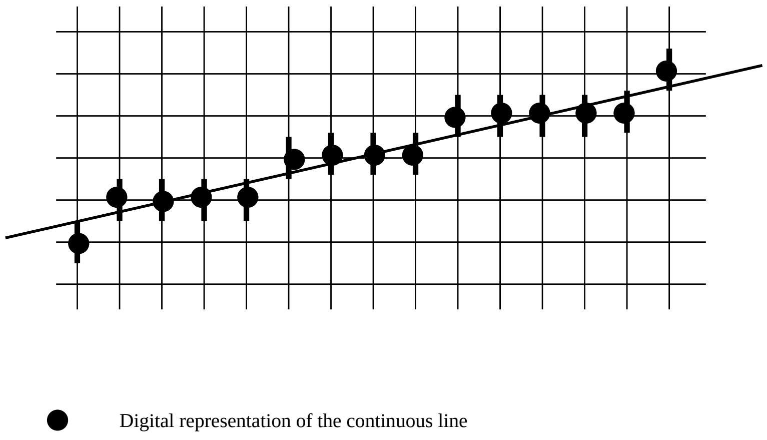


Figure 22: Digitization of a line segment through the semi-open vertical segment method.

An 8-connected path is still obtained. The digital points are distributed on both sides of the continuous line. This result is equivalent to the one we would obtain with the Bresenham algorithm, which is very popular and often used in image synthesis for instance (it minimizes the local error at each point of the digitization) [12].

4.2 Characterization of a digital line segment

Let us now consider the inverse problem: given a set of digital points, is it a possible digitization of a continuous line segment? By using the previous characteristics, answering this question amounts to verify that there exists a line segment such that its digitization according to the chosen rules provides exactly the set of digital points. This approach is not very operational and two other methods are preferred, one relying on the chord property, and the other on a syntactic description of a digital line segment.

Let $\mathcal{S}$ be a set of digital points (still in 2D and on a square grid). The set $\mathcal{S}$ is said to satisfy the chord property iff:

$$\forall (P,Q) \in \mathcal{S}, \forall R \in [P,Q],\ \exists T \in \mathcal{S}, d_\infty(T,R) < 1,$$

where $[P,Q]$ denotes the segment of $\mathbb{R}^2$ (continuous segment) linking P and Q, and d_∞ denotes the distance obtained from the L_∞ norm in $\mathbb{R}^2$ ($d_\infty((x,y),(x',y')) = \max(|x-x'|,|y-y'|)$).

Figure 23 illustrates a case where this property is satisfied, and a case where it is not.

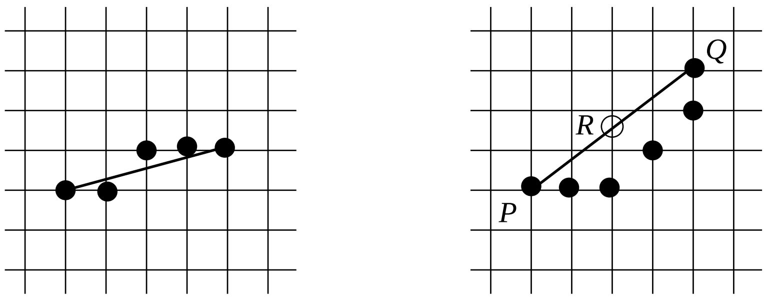


Figure 23: The set of digital points on the left satisfies the chord property, while the one on the right does not (point R on segment $[P,Q]$ is at a distance larger than 1 of every digital point of $\mathcal{S}$).

A digital segment for which the chord property holds also satisfies the criteria used in the digitization method based on semi-open segments [25, 23, 27].

A second operational characterization of digital line segments relies on their syntactic description [21]. A digital line segment consists of a series of points with specific direction changes when going from a point to the next one along the line. We have the following characterization:

- a section is a maximal sub-sequence of points without change of direction (the 8 possible directions are defined by the 8-connectivity on the square grid, as shown in Figure 24);
- sections may have only two different directions, which are consecutive (according to the schema in Figure 24);
- for one of these directions, sections have a length equal to 1, and for the other direction, sections have a length equal to n or $n+1$, where the value of n depends on the slope of the line.

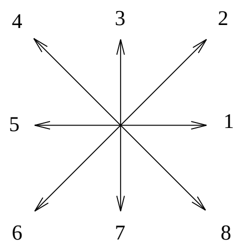


Figure 24: The 8 possible directions on a square grid. Two directions are consecutive if their label differs by 1 (modulo 8).

It follows that this characterization can be used in a very simple way in order to check whether a set of points is a digital line segment or not. This is illustrated in Figure 25.

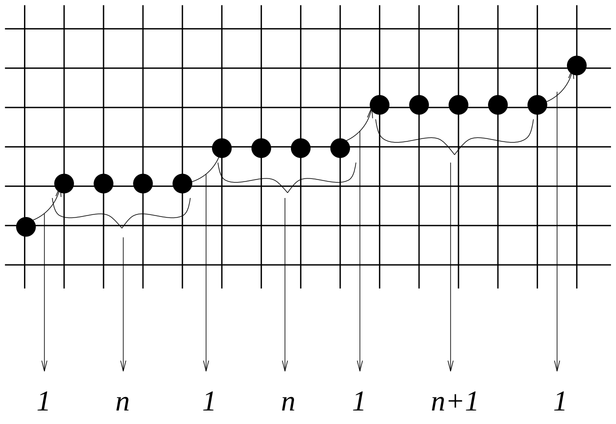


Figure 25: Syntactic characterization of a digital line segment.

4.3 Digital analytical lines

We consider now the problem of digital lines from a different point of view. Instead of trying to approximate a line by a sequence of connected digital points, we now raise the question of determining the intersection points (without approximation) between the grid and a continuous line.

A line with equation $y = ax + b$ will then be represented through its intersection points with the grid. In order to guarantee that this intersection is non empty, the slope a of the line has to verify:

$$a = \frac{p}{q},$$

with p and q are integers, mutually prime, and satisfying the following condition:

$$p \leq q \leq N,$$

for an image of size $N \times N$, and for a slope less than 1 (the other cases are deduced based on symmetry properties).

The slopes of the possible lines (having non empty intersection with the grid) constitute a Farey sequence of order N, denoted by $F(N)$ [9]. For an image of size 4×4, the possible lines such that $a \leq 1$ are represented in Figure 26; in this case we have:

$$F(N) = \{0, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\}.$$

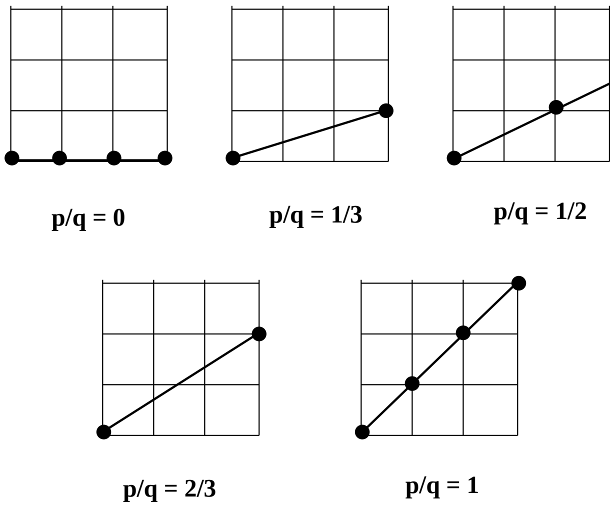


Figure 26: Possible lines (with non empty intersection with the grid points) on an image of size 4×4.

Obviously, the number of possible lines increases with the size of the image. For instance for $N = 6$, we have:

$$F(6) = \{0, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, 1\}.$$

The cardinality of $F(N)$ is of order $3N^2/\pi^2$. Some properties of Farey sequences lead to an easy recursive computation. In particular, if the fractions $\frac{p}{q}$, $\frac{p'}{q'}$, $\frac{p"}{q"}$ are consecutive terms of $F(N)$, then the following relations hold:

$$p'q - pq' = 1,$$

$$\frac{p'}{q'} = \frac{p + p"}{q + q"}.$$

This allows us to compute the number of possible lines according to the size of the image, as well as the number of occurrences of these lines as a function of slope a. From two given points in an image, some lines will be found more frequently than others. For instance lines with slope 0 and 1 in Figure 26 are obtained for more pairs of points than other lines.

For some transformations, this phenomenon can induce a strong bias, due to the digitization, that will have to be taken into account. For instance, if we try do detect straight contours by counting the pairs of points which contribute to a contour with given orientation (for instance using the Hough transform [19, 20]), contours with slope 0 or 1 will lead to higher scores and will therefore be easier to detect that contours with very low slope.

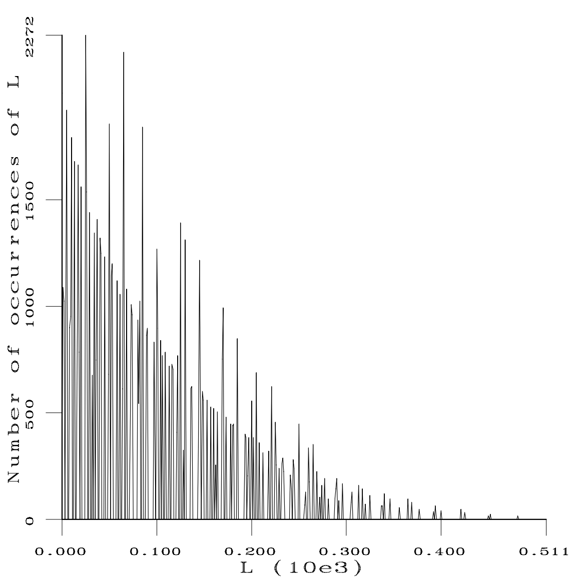
If we are now looking at the possible lengths of digital segments, conclusions of the same nature can be drawn. Let L be the squared length of a segment. The value of L is obtained as the quadratic distance between two points with integer coordinates, and is therefore solution of a Diophantine equation of the following type:

$$a^2 + b^2 = L,$$

where a and b are integers. A similar equation is obtained for digital circles, and will be examined in more details in the corresponding section.

This equation does not always have a solution. Depending on the parity of a and b, it can be observed that $a^2 + b^2$ may be congruent to 0, 1 or 2 modulo 4. The values of L which are congruent to 3 modulo 4 can never be obtained.

Figure 27 illustrates the irregularity of the numbers of occurrences of L on 16×16 and 50×50 images. The general decreasingness of the obtained curves is due to the fact that the images are bounded.



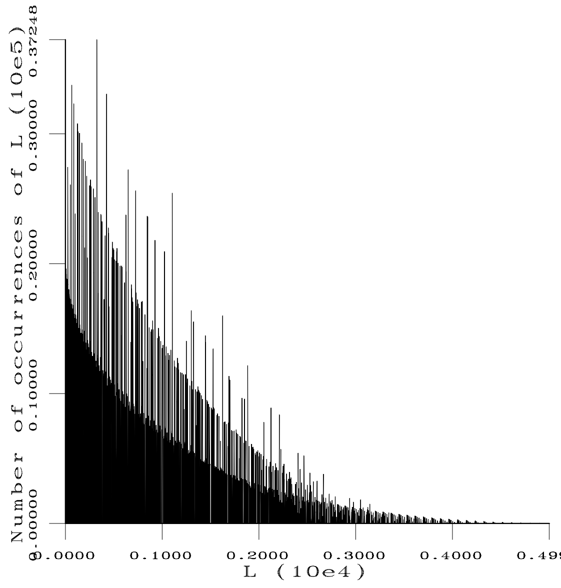


Figure 27: Distribution of the squared length of the digital segments on an image of size 16×16 and on an image of size 50×50.

For more details on digital lines and curves, the interested reader can refer to [7], to other parts of TC18 and to the associated softwares[4].

4.4 Digital circles

Similarly as for lines, the number of digital points which constitute the intersection of the grid and a continuous circle may vary a lot, and in an irregular way, depending on the radius of the circle. For some values of the radius, there is no solution at all. Here again, the solutions for the intersection problem (assuming a circle centered on a grid point) are obtained by solving the following equation:

$$a^2 + b^2 = n,$$

where a, b and n are integer values, n representing the square of the radius of the circle and $a^2 + b^2$ the quadratic distance between a digital point and the center of the circle. By solving this equation, 4 intersection points are obtained for a circle of radius 1, 4 points for $n = 2$, no point for $n = 3$, etc., as illustrated in Figure 28.

[4] `http://www.cb.uu.se/~tc18/code_data_set/code.html`

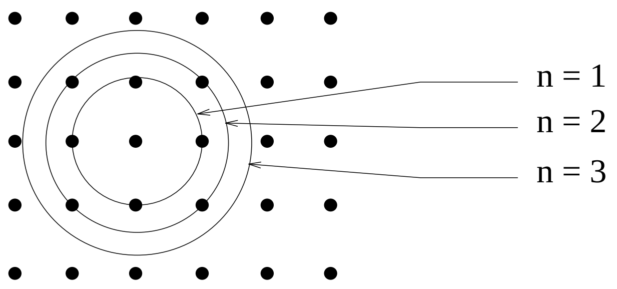


Figure 28: Intersection between continuous circles and digital points of the square grid.

The number of solutions of this equation is given by $r(n)$, computed from the decomposition of n into prime numbers as follows:

$$r(n) = 4\Pi_p(\tau_p + 1)\Pi_q\left(\frac{1 + (-1)^{\sigma_q}}{2}\right),$$

where n is decomposed into prime numbers as:

$$n = 2^\alpha \Pi_p p^{\tau_p} \Pi_q q^{\sigma_q},$$

where p denotes the prime factors congruent to 1 modulo 4 and q the prime factors congruent to 3 modulo 4.

For instance, for $n = 5$ the intersection has 8 points, for $n = 25$ it has 12 points: since $25 = 5^2$ and 5 is congruent to 1 modulo 4, the only factor in $r(25)$ corresponds to $p = 5$ with $\tau_p = 2$ and $r(25) = 4 \times (2 + 1) = 12$. As can be observed from the formula for $r(n)$, as soon as a factor congruent to 3 modulo 4 is involved in the decomposition of n with an odd exponent, then $r(n) = 0$.

Defining a digital circle only by its intersection points with the grid can appear as too restrictive, and we may prefer considering the circle as having some thickness. Note that a similar approach could also be adopted for lines and line segments by considering them as bands with some thickness.

Let k denotes the thickness. The equation to be solved becomes:

$$n \leq a^2 + b^2 < n + k,$$

where a and b are integer values. The number of solutions is then:

$$r(n, k) = \sum_{i=n}^{i=n+k-1} r(i).$$

For instance, for $n = 54$ and $k = 4$, we obtain $r(n, k) = 0$. This shows that zero values can still be obtained, even with a non neglectable thickness. Here again, the important variability of the function $r(n, k)$ induces a bias in methods for image analysis, image synthesis, pattern recognition, that should not be ignored. Some correction methods can be designed in order to account for the shape of the obtained distributions.

More details on digital circles and spheres can be found e.g. in [1].

4.5 Voronoï tessellation and Delaunay triangulation

In this Section, we briefly mention a few interesting aspects about Voronoï tessellations. They constitute important structured representations that may be adapted to the image content (as opposed to pixel-based graph representations for instance), and are therefore well suited for shape interpretation and recognition. Another example is the skeleton, which was developed in particular within the mathematical morphology framework [28].

The following is but a summary. More details are likely to be found in other parts of TC18.

Let $\{P_1, P_2, ..., P_n\}$ be a set of points, called seeds. A portion of plane is associated (in $\mathbb{R}^2$) to each of this seed, which is defined as:

$$V(P_i) = \{P \in \mathbb{R}^2 \; / \; \forall j, 1 \leq j \leq n, \; d(P, P_i) \leq d(P, P_j)\},$$

where d denotes a distance defined in $\mathbb{R}^2$. This means that $V(P_i)$ is composed of all points of $\mathbb{R}^2$ which are closer to P_i than to any other seed.

If d is taken as the Euclidean distance, then the $V(P_i)$ are convex polygons. Let us start with $n = 2$. The limit between $V(P_1)$ and $V(P_2)$ is the median line between both points, and each domain if therefore a half-plane. The $V(P_i)$ for any n are then intersections of half-planes, i.e. convex polygons (which are not necessarily bounded).

Figure 29 illustrates this definition (for the Euclidean distance), as well as the convexity of obtained cells.

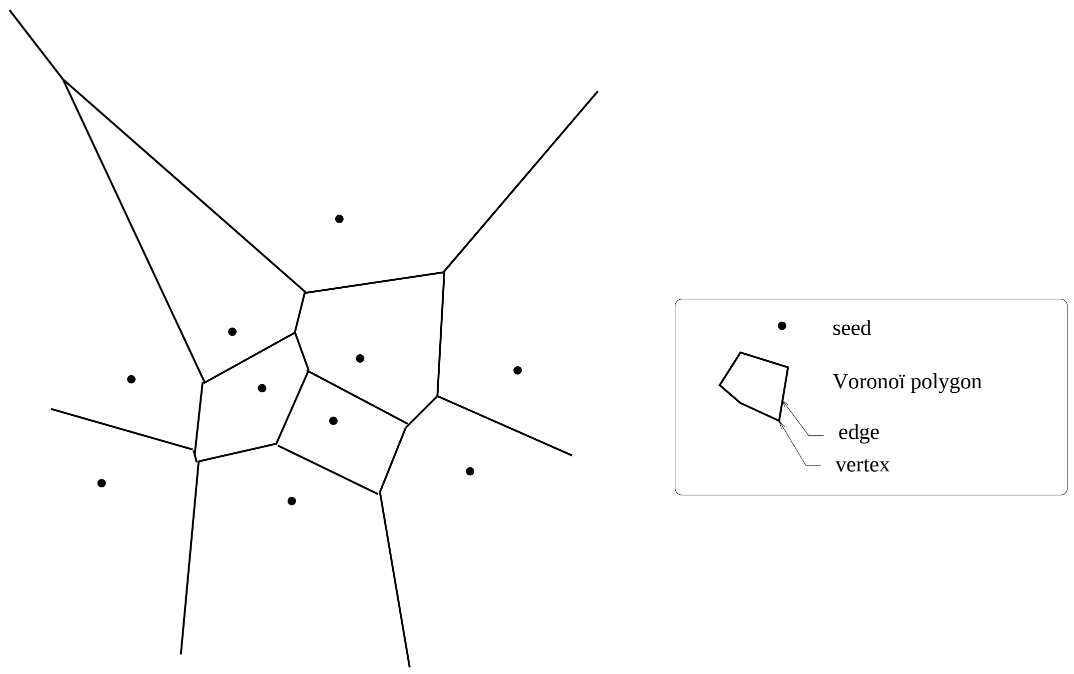


Figure 29: Voronoï tessellation from a set of points (seeds).

Let us note that the obtained tessellation corresponds to the first of the methods described in Section 2 for constructing tessellations. Cells are called Voronoï polygons and consist of Voronoï edges and vertices.

The definition of a Voronoï tessellation can be easily extended to any type of seeds (not reduced to points), leading to cells which are no more convex polygons but can have any shape. This notion is then related to the one of skeleton of influence zones [28].

In the digital case, the formal definition of Voronoï tessellation is exactly the same. However, instead of the Euclidean distance, an approximation as a digital distance is generally chosen (this question will be briefly addressed in Section 5). This leads to fast computation while providing precise enough approximations.

The Voronoï tessellation has the following properties [14]:

- if there do not exist 4-uples of cocircular seeds, then every Voronoï vertex is equidistant of exactly three seeds;
- any vertex of the Voronoï tessellation is the center of a circle going through three seeds and containing no other seed; this circle is called Delaunay circle;
- $V(P_i)$ is non bounded iff P_i belongs to the boundary of the convex hull of the set of P_j.

Now, by linking with segments the three seeds which are at equal distance of a Voronoï vertex, we obtain a triangulation of the set of points P_i, called Delaunay triangulation. The obtained result for the points in Figure 29 is illustrated in Figure 30.

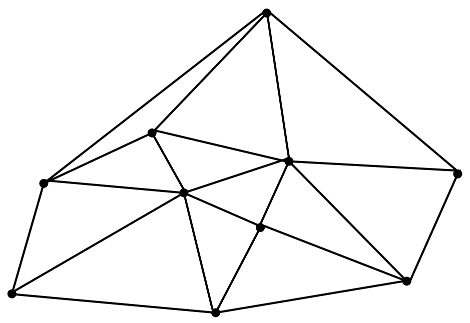


Figure 30: Delaunay Triangulation of a set of points.

Delaunay triangulation and Voronoï tessellation are two dual structural representations, as illustrated in Figure 31. A Voronoï edge corresponds to a triangulation edge, which is orthogonal to it; each seeds is a vertex of the triangulation and each Voronoï vertex if the center of a Delaunay triangle.

The second property is illustrated in Figure 32. The Delaunay triangulation is exactly the triangulation such that every circle circumscribing a triangle does not contain any vertex of the triangulation. This constitutes a first geometrical application of Voronoï tessellations and Delaunay triangulations.

The third property allows us to obtain directly the convex hull of a set of points, which constitutes a second geometrical application of these two notions.

Actually, there are many other applications. We mention one more here, concerning the computation of minimum distance between two sets of points A and B. This distance is defined as:

$$d_{\min}(A, B) = \min_{a \in A, b \in B} d(a, b),$$

where $d(a, b)$ denotes the Euclidean distance between points a and b. This minimum distance can be simply obtained by computing the Delaunay triangulation of the union of all points in $A \cup B$ and finding the shortest edge linking a points in A to a point in B.

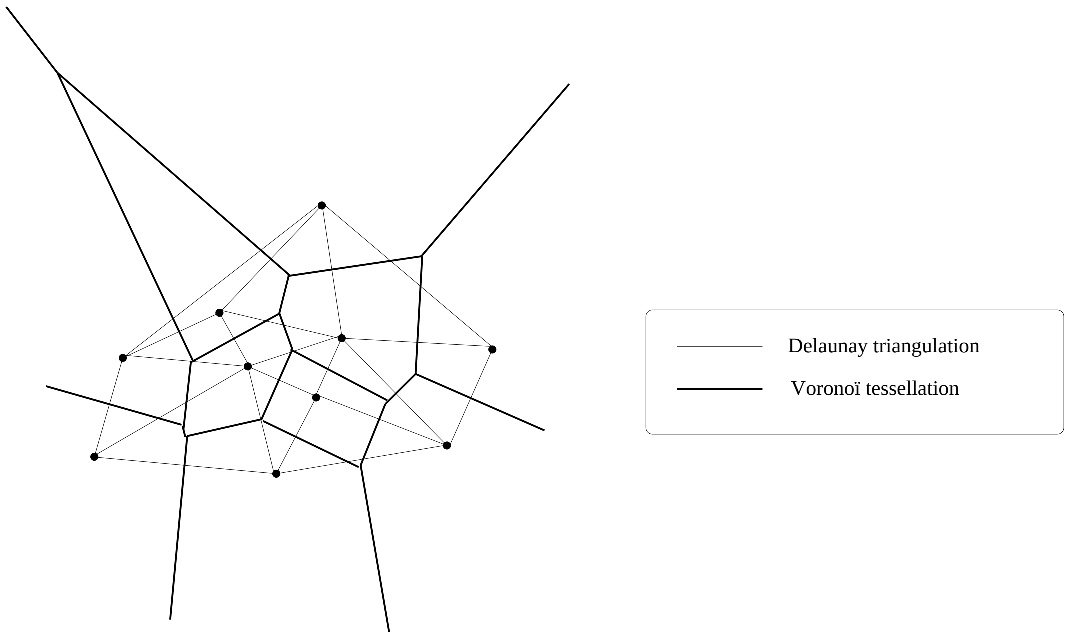


Figure 31: Duality between Delaunay triangulation and Voronoï tessellation.

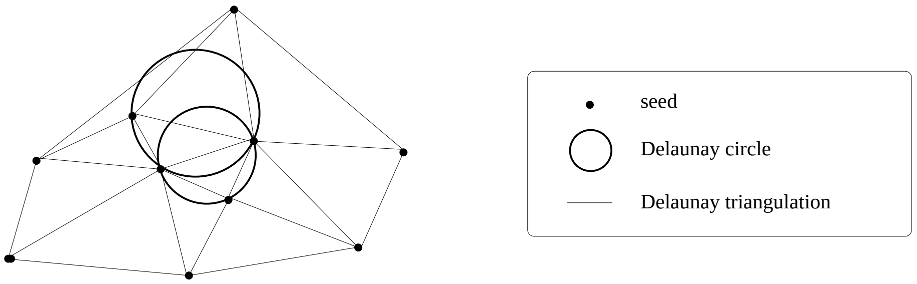


Figure 32: Geometrical properties of Delaunay triangulation.

Let us finally mention a few algorithms for computing a Voronoï tessellation [29, 22]. Some recursive algorithms are optimal in terms of algorithmic complexity, but there implementation can be tedious. On the contrary, some incremental algorithms, which may not be optimal in the worst case, may be easier to implement [10]. Let us describe one of them (see Figure 33). The idea consists in adding one point at each iteration and modifying the previous tessellation accordingly. The interest of this approach is that adding a point leads in general only to local modifications. Let us assume that the tessellation of the set of points $\{P_1, ...P_{n-1}\}$ has already been computed. When adding a new seed P_n, the seed P_i $(1 \leq i \leq n-1)$ the closest to P_n is first selected. The median line of $[P_i, P_n]$ is the support of a new edge of the tessellation. The intersection of this line with an edge of the previous tessellation is then computed. This leads to a new cell (adjacent to the initial one), and the corresponding seed becomes the new point P_i. This procedure is iterated until the initial selected seed is found again. This construction provides successively all edges defining the cell associated to P_n. The portions of edges of the previous tessellation which fall within this cell are suppressed.

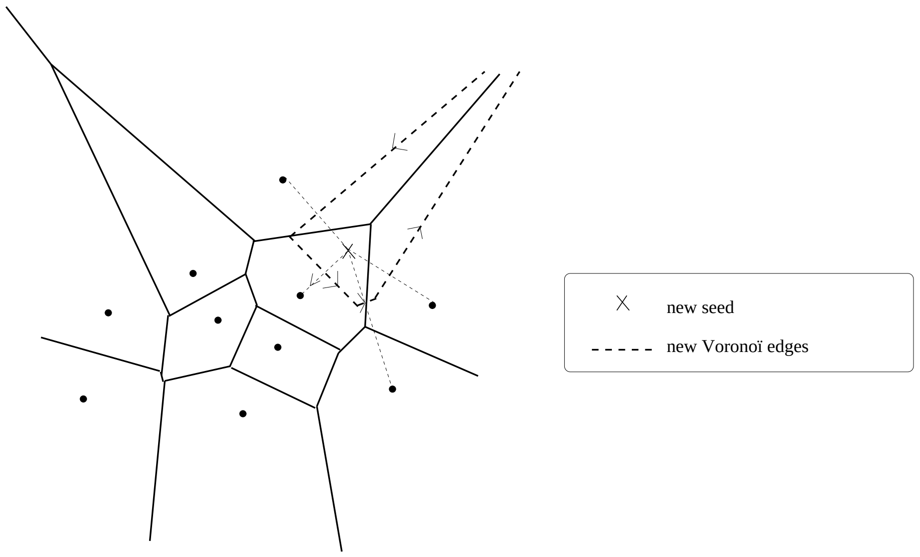


Figure 33: Incremental computation of a Voronoï tessellation.

5 Distance transform

The distance transform computes at each point of a binary digital image a value which approximates its Euclidean distance to the closest object point. Although this definition is rather global, this transform can be based on local computations only, by propagating local distances between neighbor points. These local distances are defined in a mask. This type of algorithm avoids computing the distance of a point to all object points in an exhaustive way and then finding the minimum. This would be much too long. Only an approximation of the Euclidean distance is computed this way, which should be as good as possible while keeping fast algorithms. These digital distances are called chamfer distances. They are detailed in [3] and in other sections of TC18. Only the main lines are summarized here.

The distance transform has many applications, beside just computing distances, for instance for computing fast morphological operations.

5.1 Definition of digital distances

Digital distances are defined as distances on a graph. The vertices of the graph are pixels or voxels and edges are defined from a vector basis. They can link points which are neighbors according to one of the elementary connectivities (4- or 8-connectivity for instance on a 2D square grid), or more distant points. These vectors represent directions along which it is possible to "move" to go from one point to another one, i.e. to define paths. The length of a shortest path between two graph vertices defines the distance between the corresponding points.

Let $\mathcal{P} = \{\vec{p_1}, ...\vec{p_m}\}$ the set of basis vector generating the graph. A length d_i is associated to each p_i. The following conditions have to be satisfied:

- $\vec{p_i} \in \mathcal{P} \Rightarrow -\vec{p_i} \in \mathcal{P}$;
- $\vec{p_i} \in \mathcal{P}, \lambda\vec{p_i} \in \mathcal{P} \Rightarrow \lambda = \pm 1$;
- $||\vec{p_i}|| = ||\vec{p_j}|| \Rightarrow d_i = d_j$.

The distance between two vertices x and y (two digital points) is then defined as:

$$d(x, y) = \frac{1}{s} \min\{\sum_{i=1}^{m} n_i d_i \mid n_i \in \mathbb{N}, \sum_{i=1}^{m} n_i \vec{p_i} = \vec{xy}\},$$

where s is a scale factor (typically $s = d_1$ if d_1 is the length associated to the unit vector along one of the coordinate axes). This distance is exactly the length of the shortest path between x and y on the graph generated by $\mathcal{P}$. It is easy to check that all properties of a distance are satisfied by this definition.

From a practical point of view, the $\vec{p_i}$ and d_i are represented by a mask. Each point in the mask is the end point of a vector $\vec{p_i}$ originating in the center of the mask, and has a coefficient equal to d_i, which represents the local distance between this point and the mask center (the center having a coefficient 0).

5.2 Examples

The quality of the approximation is tuned through two types of parameters:

- the size of the mask, i.e. the number of represented directions (number of vectors $\vec{p_i}$);
- and the coefficients in the mask, i.e. the values of d_i.

Several criteria have been proposed to estimate this quality. The most usual one is to minimize the maximum of the difference with the Euclidean distance.

The most usual masks are displayed in Figure 34. Examples (a) and (b) correspond to basic masks associated to the 4- and 8-connectivities on a square grid, respectively. Example (c) leads to a lower error with respect to the Euclidean distance (≤ 8 %), by modifying the coefficients (note that all distances are then multiplied by a factor 3). The last example (d) yields an error less than 2 %, by modifying both the coefficients and the size of the mask (all distances are then multiplied by a factor 5).

$$\underset{\text{(a)}}{\begin{array}{ccc} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{array}} \qquad \underset{\text{(b)}}{\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array}} \qquad \underset{\text{(c)}}{\begin{array}{ccc} 4 & 3 & 4 \\ 3 & 0 & 3 \\ 4 & 3 & 4 \end{array}} \qquad \underset{\text{(d)}}{\begin{array}{ccccc} & 11 & & 11 & \\ 11 & 7 & 5 & 7 & 11 \\ & 5 & 0 & 5 & \\ 11 & 7 & 5 & 7 & 11 \\ & 11 & & 11 & \end{array}}$$

Figure 34: Examples of masks for chamfer distance computation.

5.3 Algorithms

Once the digital distance is defined, with the required precision, the second requirement is to develop fast algorithms to compute the distance transform.

A first class of algorithms uses a classical representation of the image, as a matrix. Two types of algorithms can be found in this class: parallel procedures and sequential ones. They are described below for the above mentioned problem: computing the distance of each image point to the closest object point.

Let f^k denote the image computed at iteration k, and g the chosen mask. The p_i define the points in support(g) while the d_i define the values $g(y)$ for $y \in \text{support}(g)$. The initialization f^0 is defined as follows: the object points are set to 0 and the points of the complement are set to infinity (or a large enough value).

The parallel algorithm iterates the following equation until convergence:

$$f^k(x) = \min\{f^{k-1}(y-x) + g(y),\ y \in \text{support}(g)\}.$$

This algorithm can be applied on any type of grid and with any mask. Its main advantage is that the computation can be performed in parallel on all points. Its main disadvantage is that the number of iterations depends on the image size, on the object size and shape. Moreover, two images are needed in memory.

The sequential algorithm requires only two iterations, whatever the image and its content. It uses two opposite scanning directions. The mask is divided into two parts g_1 and g_2, containing the points already processed in the current scanning direction. For $k = 1, 2$, the following operation is applied (f^0 is defined as above):

$$f^k(x) = \min\{f^{k-1}(x), f^k(y-x) + g_k(y),\ y \in \text{support}(g_k)\}.$$

This algorithm is very fast, it requires the storage of one image only, and it applies on any grid and any mask.

An example of application of this algorithm is shown in Figure 35. The chamfer distance of the set of black dots is computed. This distance is then use to define the Voronoï tessellation associated to this set of dots.

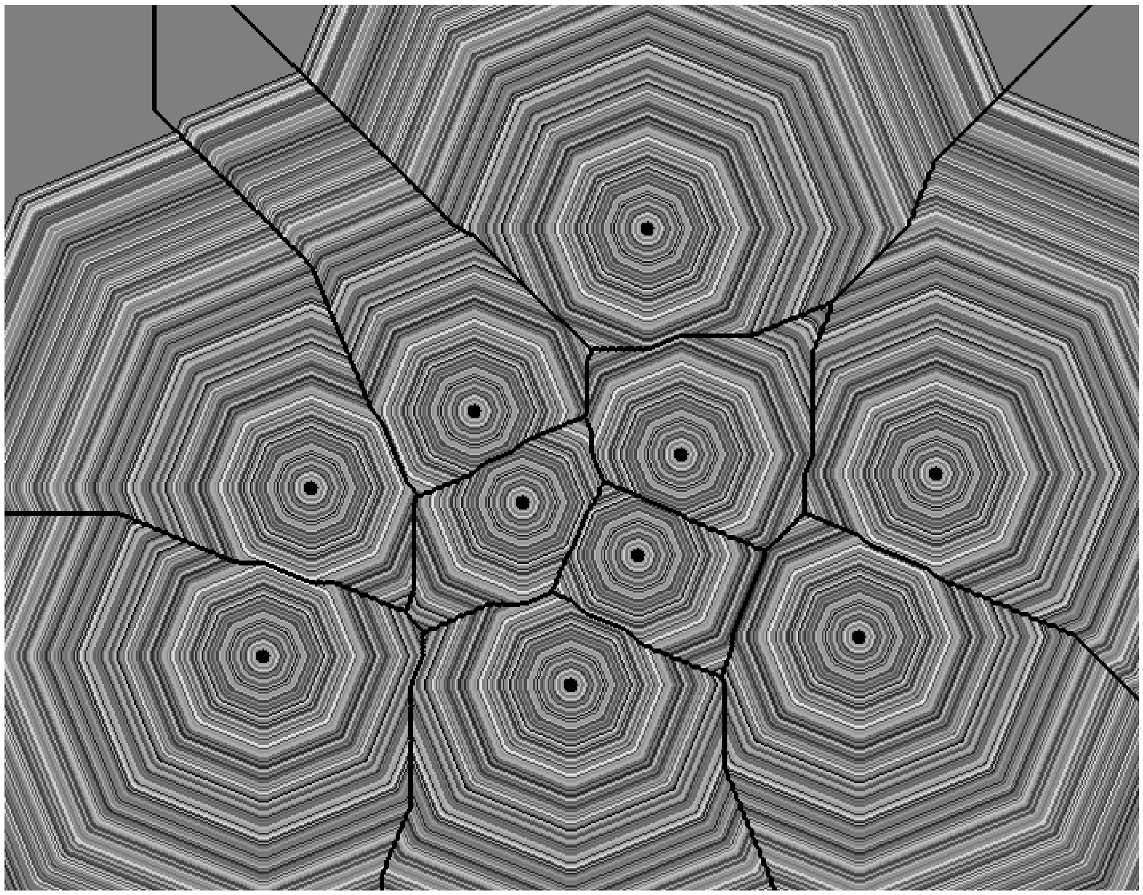


Figure 35: Chamfer distance transform of the set of black dots and derived Voronoï tessellation (black lines). For better visualization, the distance values are shown with random grey levels (the iso-distance lines appear clearly) and the edges of the tessellation have been slightly dilated.

A second class of algorithms relies on a data structure representing the object contours. This idea comes from the fact that points inside objects have always a zero distance to these objects, while points the closest to points in the complement are always situated on the boundary of the objects. Therefore only contour points are really involved in the computation.

A first type of algorithm in this class relies on a chain structure to represent boundary points. The computation relies on rules for moving these points to the points at a distance 1 of the object, then to the points at distance 2, etc. [30]. An adjustment step is then required, for instance in cases where moving contour points leads to a connection between two objects. This algorithm requires only one image in memory. The access to neighbor points should be easy within the data structure, since it is often required. The algorithm is also very fast. Rules for moving points and for adjustments need to take several particular cases into account, and the whole procedure should be carefully implemented. Unfortunately, this approach cannot be extended directly to 3D images, since there is no natural order on surface points that would lead to a chain structure.

A second type of algorithm in this class relies on a representation of contour points as a simple FIFO waiting list (*first in, first out*). This file is initialized with the contour points of the objects, without care for the order. The algorithm then extracts the first point p in the list and finds the neighbors. For each neighbor q, if it is in the complement of the objects, it it added to the list and its distance is the one of p plus the local distance between p and q (i.e. d_i if $\vec{pq} = \vec{p_i}$). This algorithm is also very fast and requires a rapid access to the neighbors as well. It can be extended to 3D.

Note that all these classes of algorithms are used for many other transformations in digital image processing.

References

[1] E. Andres. Discrete Circles, Rings and Spheres. *Computers & Graphics*, 18(5):695–706, 1994.

[2] C. Berge. *Théorie des graphes et ses applications.* Dunod, Paris, 1958.

[3] G. Borgefors. Distance Transforms in the Square Grid. In H. Maître, editor, *Progress in Picture Processing, Les Houches, Session LVIII, 1992*, chapter 1.4, pages 46–80. North-Holland, Amsterdam, 1996.

[4] J. M. Chassery and M. I. Chenin. Topologies on Discrete Spaces. In Simon and Haralick, editors, *Digital Image Processing*, pages 59–66. Reidel, 1980.

[5] J. M. Chassery and A. Montanvert. *Géométrie discrète.* Hermes, Paris, 1991.

[6] Y. Cointepas, I. Bloch, and L. Garnero. A Cellular Model for Multi-Objects Multi-Dimensional Homotopic Deformations. *Pattern Recognition*, 34(9):1785–1798, sep 2001.

[7] I. Debled-Rennesson and J.-P. Reveillès. A Linear Algorithm for Segmentation of Digital Curves. *International Journal on Pattern Recognition and Artificial Intelligence*, 9:635–662, 1995.

[8] L. Dorst and W. M. Smeulders. Discrete Representation of Straight Lines. *IEEE Trans. on PAMI*, 6(4):450–463, 1984.

[9] J. Franel. Les suites de Farey et les problèmes des nombres premiers. *Göttinger Nachrichten*, pages 198–201, 1924.

[10] P. J. Green and R. Sibson. Computing Dirichlet Tessellation in the Plane. *The Computer Journal*, 21:168–173, 1978.

[11] B. Grunbaum and G. C. Shepard. *Tilings and Patterns: an Introduction.* Freeman, 1989.

[12] G. Hégron. *Synthèse d'image : algorithmes élémentaires.* Bordas, Dunod Informatique, Paris, 1985.

[13] E. Khalimsky, R. Koppermann, and P. R. Meyer. Computer Graphics and Connected Topologies as Finite Ordered Sets. *Topology and its Applications*, 36:1–17, 1990.

[14] R. Klein. *Concrete and Abstract Voronoï Diagrams.* Springer Verlag, 1989.

[15] R. Klette and A. Rosenfeld. *Digital Geometry.* Morgan Kaufmann, San Francisco, 2004.

[16] T. Y. Kong and A. Rosenfeld. Digital Topology: Introduction and Survey. *Computer Vision, Graphics, and Image Processing*, 48:357–393, 1989.

[17] V. A. Kovalesky. Finite Topology as applied to Image Analysis. *Computer Vision, Graphics, and Image Processing*, 46:141–161, 1989.

[18] J. L. Locher, C. H. A. Broos, M. C. Escher, G. W. Locher, and H. S. M. Coxeter. *L'œuvre de M. C. Escher.* Editions du Chêne, Paris, 1972.

[19] H. Maître. Un panorama de la transformation de Hough. *Traitement du Signal*, 2(4):305–317, 1985.

[20] H. Maître. Contribution to the Prediction of Performances of the Hough Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):669–674, 1986.

[21] S. Pham. Digital Straight Segments. *Computer Vision, Graphics, and Image Processing*, 36:10–30, 1986.

[22] F. P. Preparata and M. I. Shamos. *Computational Geometry, an Introduction.* Springer Verlag, 1988.

[23] C. Ronse. A Simple Proof of Rosenfeld's Characterization of Digital Straight Line Segments. *Pattern Recognition Letters*, 3:323–326, 1985.

[24] A. Rosenfeld. Connectivity in Digital Pictures. *Journal of ACM*, 17(1):146–160, 1970.

[25] A. Rosenfeld. Digital Straight Line Segments. *IEEE Trans. on Computers*, 23(12):1264–1269, 1974.

[26] A. Rosenfeld. Digital Topology. *Amer. Math. Monthly*, pages 621–630, 1979.

[27] L. A. Santalo. Complemento a la Nota: un Teorema Sobre Conjuntos de Paralelepipedos de Aristas Paralelas. *Publ. Int. Mat. Univ. Nac. Litoral*, 2:49–60, 1940.

[28] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, New-York, 1982.

[29] G. T. Toussaint. *Computational Geometry.* North Holland, Amsterdam, 1985.

[30] L. Vincent. Morphological Algorithms. In E. Dougherty, editor, *Mathematical Morphology in Image Processing*, pages 255–288. Marcel Dekker, 1992.